Cleaning Your Wrong Google Scholar Entries

Shuang Hao† Yi Xu† Nan Tang‡ Guoliang Li† Jianhua Feng†
†*Department of Computer Science, Tsinghua University, China* ‡*Qatar Computing Research Institute, HBKU, Qatar*
{*haos13@mails., xuyi15@mails., liguoliang@, fengjh@*}*tsinghua.edu.cn, ntang@hbku.edu.qa*

***Abstract*—Entity categorization – the process of grouping entities into categories for some specific purpose – is an important problem with a great many applications, such as Google Scholar and Amazon products. Unfortunately, many real-world categories contain mis-categorized entities, such as publications in one's Google Scholar page that are published by the others. We have proposed a general framework for a new research problem – *discovering mis-categorized entities*. In this demonstration, we have developed a Google Chrome extension, namely `GSCleaner`, as one important application of our studied problem. The attendees will have the opportunity to experience the following features: (1) *mis-categorized entity discovery* – The attendee can check mis-categorized entities on anyone's Google Scholar page; and (2) *Cleaning onsite* – Any attendee can login and clean his Google Scholar page using `GSCleaner`. We describe our novel *rule-based framework* to discover mis-categorized entities. We also propose effective optimization techniques to apply the rules. Some empirical results show the effectiveness of `GSCleaner` on discovering mis-categorized entities.**

***Keywords*-mis-categorized entity; Google Scholar cleaner; rule-based framework; signature**

I. Introduction

Categorizing entities into sensible groups is a fundamental mode for many applications, such as one's publication records in Google Scholar or DBLP, and product categories in Amazon or eBay. Unfortunately, mis-categorized entities are everywhere, *e.g.,* there exist others' publications in many researchers' Google Scholar pages – the same issues happen in other sources such as Scopus and Web of Science. The above scenarios motivate a new research problem, namely *discovering mis-categorized entities*, which is to find the entities that should not belong to a given group of entities.

Entity categorization (EC) is related to, but different from, entity matching (EM) that has been well tamed by database researchers. EM is to determine whether two objects are the same, which is often solved by comparing aligned attributes symbolically (or syntactically). In contrast, EC groups entities more conceptually (or semantically). Not surprisedly, traditional EM solutions do not well suit EC.

In fact, discovering mis-categorized entities is hard – all entities that are categorized into the same group already use all attributes of the entities, how can we use the same set of attributes to discover mis-categorized entities? Intuitively, since entities are categorized more conceptually than symbolically, we need to capture semantic similarity between entities, which is beyond traditional symbolic reasoning widely used in EM.

Our Methodology. We have proposed a new methodology of discovering mis-categorized entities – a *rule-based framework* in an easy tunable way, where the users do not need to know the back-end algorithms. The core of the framework is to first use conservative *positive rules* to identify disjoint partitions that all entities within the same group should be categorized together. Based on the assumption that the largest partition for one category should be correct, the partition with the largest size is called the *pivot partition*. Pivot partition is used collaboratively to compare with other partitions using *negative rules* to identify mis-categorized entities. We shall further elaborate our solution in Section III.

Cleaning Google Scholar. This demo instantiates our framework of discovering mis-categorized entities using Google Scholar – an important application that is relevant to almost all ICDE attendees. Google scholar data is known to be dirty[2], since it targets to cover 80-90% of all articles published in English [5]. To achieve the above coverage, Google Scholar uses Googlebots[3] that are Google's web-crawling robots to collect documents from the web. Google has done a lot of work to differentiate scholar pages to the other Web pages. Unfortunately and not surprisedly, there are still many erroneous Web pages. Moreover, the data automatically extracted from Web pages is hard to ensure cleanliness, since the Web pages might already be dirty themselves.

Challenges. Although important, removing erroneously listed papers is currently *not* an easy task. Firstly, the Google Scholar pages of some people contain many publications. For example, there are 3000 publications in Jeffrey Xu Yu's page. It is tedious and hard to examine all his entries. Secondly, there are many types of errors, e.g., string matching error, ontology matching error, etc.

Note that EC is different from Named Entity Disambiguation (NED), the task of disambiguating named entities mentioned in text of data and link them to their corresponding entries [7]. Furthermore, although there are many clustering

[1]Guoliang Li is the corresponding author.

[2]http://blog.impactstory.org/googe-scholar-profiles-fail
[3]http://www.google.com/bot.html

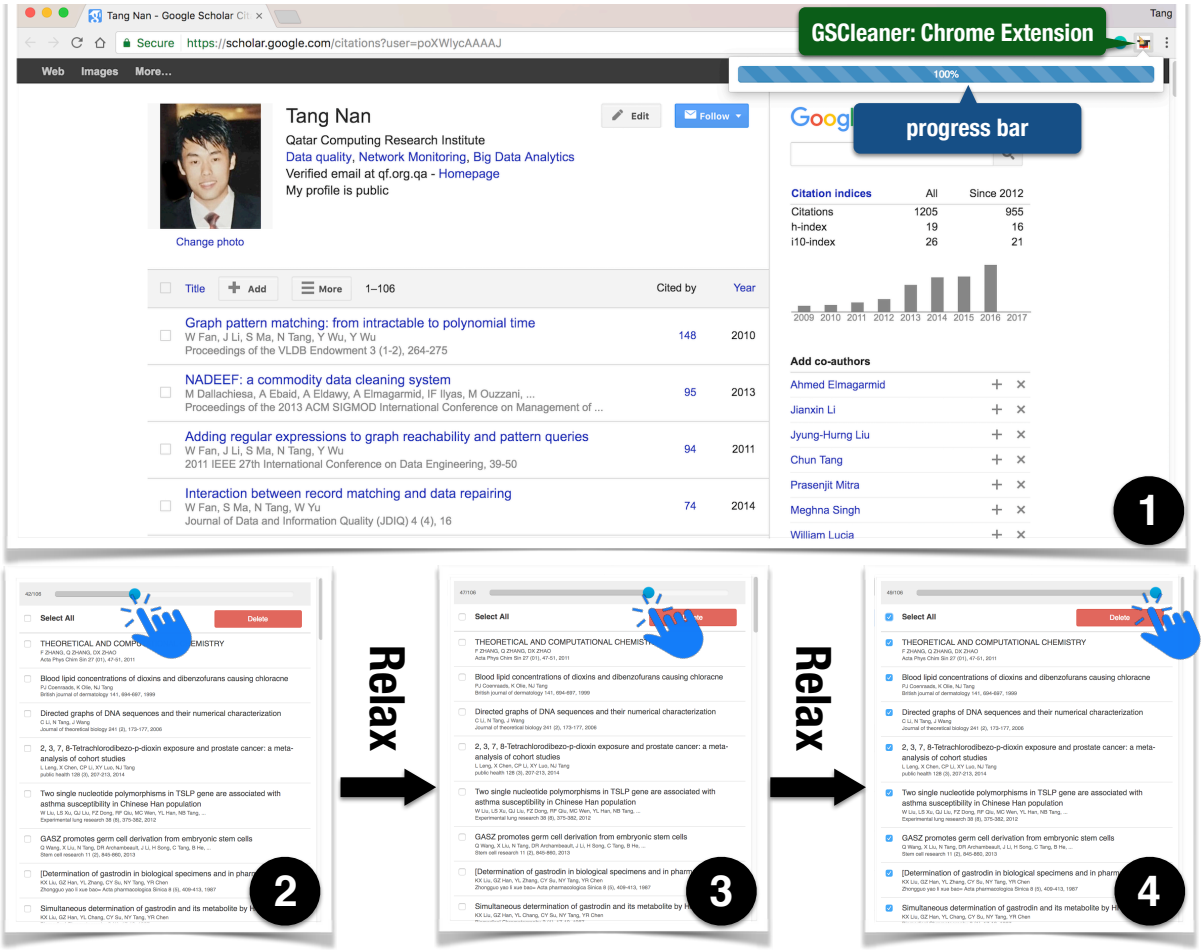

Figure 1. Clean Nan's Google Scholar Page

algorithms [3], apparently, finding two perfect clusters for close entities (categorized in the same group), one for correct entities and the other for mis-categorized entities, will fail.

II. Demonstration Overview

We have implemented a Google Chrome extension called `GSCleaner`, which has been released on GitHub[4]. Next, we will show how one can easily use `GSCleaner` to clean your wrong Google Scholar entries.

(i) Installation. One can download `GSCleaner` from GitHub, follow the instruction to install it on Google Chrome, and `GSCleaner` will show up in the top right corner of a Chrome browser, as shown in Figure 1.

(ii) Data Preparation. In anyone's Google Scholar page, one can click `GSCleaner`, and it will download the required data, whose progress is monitored by the "progress bar" as shown in the part ❶ of Figure 1. It will take approximately 2 seconds to download 100 entries.

[4]https://github.com/TsinghuaDatabaseGroup/googlescholar

(iii) Cleaning Your Google Scholar. One fundamental question is: *Will a fully automatic algorithm exist, such that no parameter tuning is needed for finding mis-categorized entities given an arbitrary group of entities?* The intuitive answer is negative, by following the *one doesn't fit all* philosophy. We propose to use a simple scrollbar to slightly relax the condition of discovering mis-categorized entities.

After the data has been downloaded, the "*scrollbar*" is activated and the user can drag it to compute different results, as shown in part ❷ of Figure 1. Our scrollbar was designed in the way that, when scrolling from left to right, the number of discovered mis-categorized entities will increase monotonically. This is done by gradually relaxing the conditions, running in the back-end, to find those entities.

As shown in Figure 1 parts ❷, ❸ and ❹, when one drags the scrollbar from left to right, the number of discovered mis-categorized entities change from 42, 47, to 49, for Nan.

When logging in, one can select the entities he wants to delete from his Google Scholar page, as shown in Figure 1 part ❹. Once confirmed, the selected entities will be removed from his Google Scholar page.

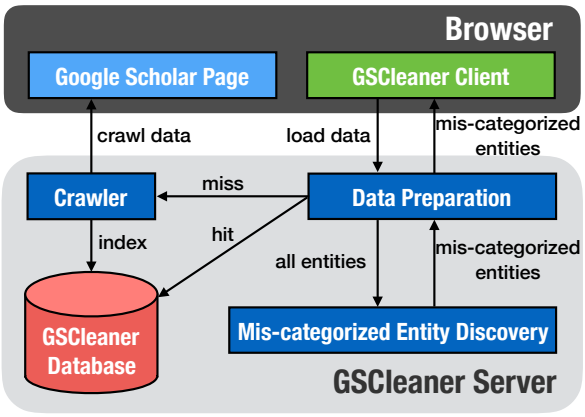

Figure 2. System Architecture

III. A Rule-based Framework

Next we discuss the architecture of GSCleaner and our rule-based framework to discovering mis-categorized entities.

System Architecture. The architecture of GSCleaner is depicted in Figure 2. When a user logs in his Google Scholar page and opens our Chrome extension, the GSCleaner client will send a request to the *Data Preparation* module in the server to get his publication entities. If his publication entities are not in our database or have been updated, it will start up a *Crawler* to extract his entities from the Google Scholar page. Otherwise, it reads the data from the database. After all data has been prepared, the *Mis-categorized Entity Discover* module will be invoked to detect all mis-categorized entities and return to the client. The user can select all or some recommended entities to clean up his Google Scholar.

Positive Rules and Negative Rules. We use positive and negative rules to discover mis-categorized entities. A *positive rule* $\varphi^+(e, e')$ is a *conjunction* of predicates: $\varphi^+(e, e') = \bigwedge_{A_i \in R} f_i(A_i) \geqslant \theta_i$, where $f_i(A_i)$ is a similarity function and θ_i is a threshold. $\varphi^+(e, e')$ is evaluated to be *true* if all predicates $f_i(A_i) \geqslant \theta_i$ return *true*. A *negative rule* $\phi^-(e, e')$ is defined similarly: $\phi^-(e, e') = \bigwedge_{A_i \in R} f_i(A_i) \leqslant \sigma_i$.

We support three types of similarity functions to quantify the similarity between two values on attribute A:

(i) Set-based. It first splits each value into a set of tokens and then utilizes the set-based similarity to quantify the similarity, such as overlap and Jaccard similarity [6].

(ii) Character-based. It measures the similarity between two values based on character transformations, like edit distance.

(iii) Ontology-based. Ontology is usually modeled by a tree structure, *e.g.,* the ontology for venues of publications provided by Google Scholar Metric[5] is shown in Figure 3. Given two entities e and e', suppose their mapping nodes on the ontology tree are n and n', respectively. Their *ontology*

[5]https://scholar.google.com/citations?view_op=top_venues.

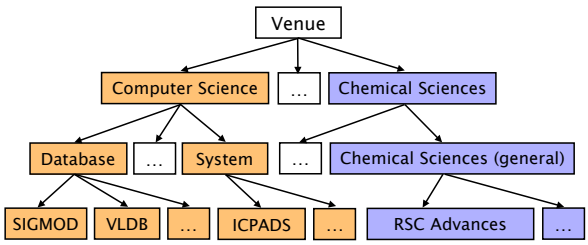

Figure 3. Google Scholar Metrics

similarity is defined as: $\frac{2|\mathtt{LCA}(n,n')|}{|n|+|n'|}$, where $\mathtt{LCA}(n, n')$ is the lowest common ancestor of n and n', and $|n|$ is the depth of node n in the tree [8]. Two entities are *similar* if their similarity is larger than a threshold τ. *e.g.,* the ontology similarity of two nodes *SIGMOD* and *VLDB* in Figure 3 is $\frac{3}{4}$, because their depth is 4 and their LCA is *Database* with depth 3.

Mis-categorized Entity Discovery. We are now ready to introduce our algorithmic framework to discover mis-categorized entities. Let $\mathbf{G} = \{e_1, \ldots, e_n\}$ be a group of entities, $\Sigma^+ = \{\varphi_1^+, ..., \varphi_x^+\}$ be a set of positive rules (*e.g.,* if two entities share $\geqslant 2$ common authors, the two entities should be in the same category), and $\Sigma^- = \{\phi_1^-, ..., \phi_y^-\}$ be a set of negative rules (*e.g.,* if two entities have $\leqslant 1$ common author, the two entities should be in different categories). Also, we assume the *transitivity* of matched entities: If e matches e' and e' matches e'', then e and e'' match. The solution overview is given in Figure 4. Here, a red circle represents a mis-categorized entity.

Step 1. [Computing Disjoint Partitions.] It applies a set of positive rules Σ^+ as a disjunction (*i.e.,* $\varphi_1^+ \vee \cdots \vee \varphi_x^+$) on $\mathbf{G}$ to group entities into partitions. Two cases are considered to put entities e and e' in the same partition. (i) e and e' satisfy a positive rule, *i.e.,* there exists a positive rule φ^+ such that $\varphi^+(e, e')$ returns true. (ii) e and e' satisfy transitivity – there exists an entity e'' such that both (e, e'') and (e', e'') satisfy some positive rule.

Step 2. [Identifying the Pivot Partition.] The pivot partition $P^* \in \mathbf{P}$ is the one with the largest size, which is taken as the correctly categorized group.

Step 3. [Discovering Mis-categorized Entities.] Given the set $\mathbf{P}$ of partitions and the pivot partition P^*, we use the negative rules to mark whether other partition $P \in \mathbf{P} \backslash \{P^*\}$ is a wrongly categorized partition, such that if P is, all entities in P should be mis-categorized. To discover the mis-categorized partitions, we enumerate every entity pair $(e^* \in P^*, e \in P)$, and every negative rule $\phi^- \in \Sigma^-$, if $\phi^-(e, e^*)$ returns true, we mark P as a wrongly categorized partition.

Tuning Negative Rules. It is impossible to use one negative rule for all groups. Hence, we provide a simple way for the user to tune the negative rules. We either apply the first rule ϕ_1^-, or jointly use ϕ_1^- with the other negative rules in sequence as $\phi_1^- \vee \phi_2^-$, $\phi_1^- \vee \phi_2^- \vee \phi_3^-$, and so on, to discover

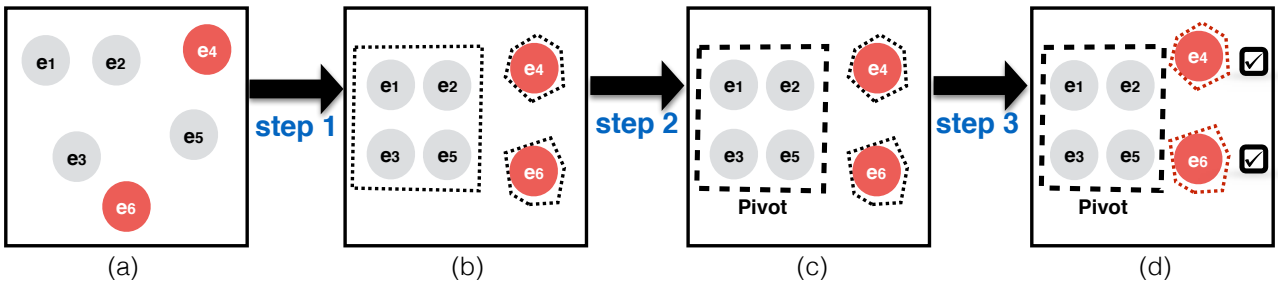

Figure 4. Solution Overview

mis-categorized entities.

Rule Generation. We propose to use positive and negative examples to generate high-quality rules and the details are referred to our paper [2].

IV. Signature-based Fast Solution

The naïve method of applying positive/negative rules is by enumerating all pairs of entities and every rule – clearly an expensive solution. In order to provide timely response, we propose a signature-based fast solution to efficiently find entity pairs satisfying given positive or negative rules.

Signatures of Entities. Signatures are substrings (or ancestor nodes) of entities that can be used to prune pairs of entities that cannot satisfy a positive/negative rule. If two entities satisfy a positive rule, they must share enough common signatures. For set-based and character-based functions, we use existing techniques to generate signatures [4]. For ontology similarity, given a node n, let $\tau_n = \lceil \frac{\theta|n|}{2-\theta} \rceil$ and denote $\mathcal{A}_{\tau_n}$ by the ancestor of n at depth τ_n. We can take $\mathcal{A}_{\tau_n}$ as a signature of n. To generate the signature for all nodes, we set τ_{min} as the minimum depth of their signatures. Then for each node n, we select $\mathcal{A}_{\tau_{min}}$ as its node signature and use this signature to find similar entities. More details can be found in our paper [2].

Efficient Algorithms for Positive Rules. The "*filter*" step: we first generate signatures for each entity *w.r.t.* each positive rule. Then, for each rule φ_i^+, we build an inverted index of the signatures of all entities, where each inverted list maintains a mapping from a signature to a list of entities that contain the signature – $L_{\varphi_i^+}(sig)$ contains the set of entities with sig as a signature. Then every entity pair (e, e') on $L_{\varphi_i^+}(sig)$ is a candidate pair. Two entities that are not on the same inverted list cannot be similar – they do not share any common signature. The "*verification*" step: we utilize transitivity to avoid verifying unnecessary candidates: given a candidate pair, we check whether they are in the same connected components. If so, we do not need to verify them. Note that the order of verifying the candidates will also affect the performance. Wang et al. [9] proved that it is optimal to ask the candidate pairs sorted by the probabilities in descending order. Thus, we will first testify the entity pairs that have lower computational cost and higher similar probability.

Efficient Algorithms for Negative Rules. To effectively check whether two partitions are dissimilar, we also utilize the signatures in the "*filter*" step: if two entities share common signatures, they may be similar; however, if two entities do not share any signature, they must be dissimilar. So, we generate the signature of a partition which is the union of signatures of entities in the partition. If two partitions have no common signatures, they satisfy the negative rule; otherwise, we need to verify the pairs of entities across these two partitions. The "*verification*" step: we first verify the entity pair with higher probability to be dissimilar, as once we find a pair satisfying the negative rule, we do not need to verify the other pairs.

Acknowledgment

This work was supported by the 973 Program of China (2015CB358700), NSF of China (61632016, 61472198, 61521002, 61661166012), and TAL education.

References

[1] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *TKDD*, 1(1):5, 2007.

[2] S. Hao, N. Tang, G. Li, and J. Feng. Discovering mis-categorized entities. In *ICDE*, 2018.

[3] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[4] Y. Jiang, G. Li, J. Feng, and W. Li. String similarity joins: An experimental evaluation. *PVLDB*, 7(8):625–636, 2014.

[5] M. Khabsa and C. L. Giles. The number of scholarly documents on the public web. *PLOS ONE*, 2014.

[6] G. Li, D. Deng, J. Wang, and J. Feng. PASS-JOIN: A partition-based method for similarity joins. *PVLDB*, 5(3):253–264, 2011.

[7] Y. Li, C. Wang, F. Han, J. Han, D. Roth, and X. Yan. Mining evidences for named entity disambiguation. In *SIGKDD*, pages 1070–1078, 2013.

[8] Z. Shang, Y. Liu, G. Li, and J. Feng. K-join: Knowledge-aware similarity join. *IEEE Trans. Knowl. Data Eng.*, 28(12):3293–3308, 2016.

[9] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, 2013.